DEPARTMENT OF AGRICULTURAL ECONOMICS & RURAL SOCIOLOGY
The Ohio State University
2120 Fyffe Road
Columbus, Ohio 43210

Prepared for discussion at
The Waste Disposal Workshop
Purdue University
April 20-21, 1971

Economics and Sociology
Occasional Paper No. 77

THE LIMITATIONS OF DYNAMIC PROGRAMMING IN WATER SYSTEM PLANNING AND THE USE OF DDDP AND GP

by
Terry Glover

Department of Agricultural Economics and Rural Sociology
The Ohio State University
2120 Fyffe Road
Columbus, Ohio 43210

THE LIMITATIONS OF DYNAMIC PROGRAMMING IN WATER SYSTEM PLANNING AND THE USE OF DDDP AND GP[1/]

I. Introduction

Dynamic programming (DP) has been used as a tool of analysis in water resource planning for some time now (C. F. Maas et al. [16], Kerri [11], Chow and Merideth [6], Hall et al. [9], Burt [5], Gablinger and Loucks [8] and others). The interest in dynamic programming has developed because it is applicable as a tool to handle multi-stage processes involved in water system design and planning. The stages involved in water resource planning may represent different points in space (pipeline routing), or they may represent different points in time (reservoir release). The stages are usually finite in number and the decision of each stage must be transformed into a state associated with the next stage. Dynamic programming as based on Bellman's [1] principle of optimality, that an optimal policy has the property that remaining decisions successive to the initial state and decision constitute an optimal policy with regard to the state resulting from the first decision, becomes an appropriate multi-stage and decision transforming tool in planning decisions for water systems and allocation.

Despite the appropriateness of DP in multi-stage analysis some serious limitations arise in its use. The purpose of this paper is to briefly review DP and the areas of concern regarding its use in applied research, particularly in water system planning, and to suggest two alternative approaches for solving multi-stage problems both of which are not entirely new techniques but have not been fully utilized in light of the limitations

---

1/ Helpful comments were received from V. T. Chow. All errors, of course, remain the responsibility of the author.

of DP. These alternative approaches are discrete differential dynamic programming (DDDP), and geometric programming (GP) or nonlinear decomposition as the GP technique is sometimes referred to. The second section of the paper gives a review of DP while the third section gives a basic formulation of DDDP. The fourth section gives a basic formulation of the geometric programming problem and its relation to nonlinear decomposition. A brief example is then explained in the final section illustrating the use of GP in a water quality problem. In the interest of brevity and due to short preparation for this paper, an example illustrating DDDP is not given.

II. Dynamic Programming and Its Limitations

The most common DP algorithm begins with the computation of the final stage and works backward toward the initial stage, given initial state values. For water system studies it is more useful to use a procedure called the forward DP algorithm. The computations are then interpreted in suitable manner.

Consider the dynamic equation of a discrete system in the time interval $t_0 \leq t \leq t_e$, $t \in [t_0, t_e]$. Segment this interval into N equal segments of small length Δt. Consider for each time interval (stage), only discrete values of states and decisions initially, then the following difference equation describes the dynamics of the discrete system:

$$(1) \quad s(n) = \phi [S(n-1), u(n-1), n-1] \quad \text{for } n = 1, 2, \ldots\ldots, N$$

where,

n = index of the stage variable,

$s(r)$ = an m-dimension state vector at stage n,

$u(n)$ = an 1-dimensional decision vector at stage n

$u(n)$ transforms the state of the system from $s(n)$ to $s(n+1)$, and

$$s(n) \in S(n)$$

$$u(n) \in U(n)$$

Where $S(n)$ is the admissible domain in the state space at stage n, and $U(n)$ is the admissible domain in the decision space at stage n. From equation (1) we derive:

$$(3)\quad s(n-1) = \phi\ \underline{/}\, s(n),\ u(n-1),\ n-1 \,\underline{/}$$

If the state of the discrete system at stage $n = o$ is $g(o)$, application of a sequence of decision vectors to this system in the time span between $n = o$ and $n = N$ will transform the state of the system to some $s(N) \in S(n)$ at stage N and produce a measure of effectiveness of policy; i.e., a return $E\underline{/}\, s(N),\ N \,\underline{/}$.

Let the objective criterion be such that the return from the system is to be maximized yielding the objective function,

$$(4)\quad \text{Max } E\underline{/}\, s(N), N \,\underline{/} = \sum_{n=1}^{N} H\underline{/}\, s(n-1),\ U(n-1),\ n-1 \,\underline{/}$$

$E\underline{/}\, s(N),\ N \,\underline{/}$ is the sum of the returns which result from series of transformations from some initial state ($n = o$) to some ending state ($n = N$). $H\underline{/}\, s(n-1),\ u(n-1),\ n-1 \,\underline{/}$ is the return from the system due to the system being in state $s(n-1)$ at stage $n-1$ and the application of a decision vector $u(n-1)$ in the time interval starting at stage $n-1$ and running for time Δt.

Assume the initial values of $E^* \underline{/}\, s(o),\ o \,\underline{/}$ (where E^* is optimal) $\forall\ s(o) \in S(o)$ are performed for a stage representing $t = t_o + \Delta t$ $(n = 1)$. Equation (4) may be written as:

$$(5)\quad E^* \underline{/}\, s(1),\ 1 \,\underline{/} = \max_{u(o) \in U(o)} E\underline{/}\, s(1),\ 1 \,\underline{/}$$

which is equal to:

$$(6)\quad \underset{u(o)\in U(o)}{\text{Max}} \left\{H\left[\bar{s}(o),\ u(o),\ o\bar{]} + E^*\left[\bar{s}(o),\ o\bar{]}\right\}\right.$$

for time $E^*\left[\bar{s}(1),\ 1\bar{]}\right.$. Now substitute equation (3) into equation (6) to obtain,

$$(7)\quad E^*\left[\bar{s}(1),\ 1\bar{]}\right. = \underset{u(o)\in U(1)}{\text{Max}} \left\{H\left[\bar{\ominus},\ u(o),\ o\bar{]} + E^*\left[\bar{\ominus},\ o\bar{]}\right\}\right.$$

which for every discrete level of state at n - 1, s(1), may be solved as a function only of u(o). The first admissible state domain at n = 1, S(1), is made discrete into d_i levels in the i-th component of the state vector, i - 1, 2, ..., m, and the admissible decision domain at n = o, U(o), is segmented into Q_j discrete levels in the j-th component of the decision vector, j = 1, 2, ..., l. Now, a point (lattice point), s(1), in the discrete state domain may be chosen and all of the admissible discrete levels of the decision vector may be applied to this level of the state vector to determine which decision vector maximizes equation (7). The first term on the right-hand side of equation (7) is calculated directly for each $u(o) \in U(o)$. The second term on the right-hand side of equation (7) is obtained by interpolation in $E^*\left[\bar{s}(o),\ o\bar{]}\right.$. The values of the sums for the various u(o) are compared to determine the maximum. The procedure is repeated for each discrete value of s(1).

The calculations may be performed for stage n = 2 representing $t = t_o + 2\,\Delta\, t$. This procedure goes through similar steps but the procedure is compounded with two decision vectors which must be considered in sequence. The decision vector applied to the system in the time span between stages n - o and n = 1 is considered in sequence with the decision vector applied between stages n = 1 and n = 2. The maximum of equation (4) for n = 2 is:

$$(8)\quad E^*\left[\bar{s}(2),\ 2\bar{]}\right. = \underset{u(1)\in U(1)}{\max} \left\{H\left[\bar{s}(1),\ U(1),\ 1\bar{]} + E\left[\bar{s}(1),\ 1\bar{]}\right\}\right.$$

(equation continued)

$$= \max_{7(1) \in U(1)} \left\{ H\left[\phi \left[s(2), u(1), 1 \right], u(1), 1 \right] + E^*\left[\phi \left[s(2), u(1), 1 \right], 1 \right] \right\}$$

Equation (8) is solved as a function of only $u(1) \in U(1)$ for every discrete level of the state vector at n = 2. This is done as was accomplished above with n = 1 and n = 2 replacing n - 0 and n = 1 respectively.

The computations may continue in similar manner to n stages. The general form of the equation for state s(n) at stage n is called the recursive equation (sometimes called functional equation) of dynamic programming. This recursive equation is written:

$$(9)\quad E^*\left[s(n), n \right] = \max_{u(n-1) \in U(n-1)} \left\{ H\left[s(n-1), u(n-1) \right] + E^*\left[s(n-1), n-1 \right] \right\}$$

At the end of the analysis an evaluation of the entire process can be made. A trace can then be made from state W to state 0 to retrieve the optimal policy satisfying specific initial and final states. This policy is obtained from among optimum decision vectors previously determined for each state at each stage. An optimal trajectory is defined by introducing the decision vectors of the optimum policy into the system equations to obtain the related optimum states.

The introduction of random elements into equation (9) is accomplished in a clear manner. A random disturbance, r(n - 1), is introduced and affects the state of the system at any stage, n, as well as do decisions u(n - 1). Equation (1) then becomes:

$$(10)\quad s(n) = \phi\left[s(n-1), u(n-1), r(n-1), n-1 \right]$$

The introduction of the random disturbances transforms s(n) into a random variable. A new objective function is also derived in that the mathematical expectation of the returns of the system is now maximized and can be written:

$$(11)\quad \max E\lfloor \bar{s}(N), N \rfloor \bar{} = E_v \left\{ \sum_{n=1}^{N} H\lfloor \bar{s}(n-1), u(n-1), r(n-1), n-1 \rfloor \bar{} \right\}$$

where $E_v \{ \cdot \}$ denotes the expected value of the terms within the outer brackets.

Assuming that the disturbances at stage n, n - 1, and n + 1 are independent of each other and the probability density function for r(o), r(1), ..., 4(n - 1) is known for y discrete levels in the range, $-\infty$ to $+\infty$, the recursive equation (9) may be written as,

$$(12)\quad E^*\lfloor \bar{s}(n), n \rfloor \bar{} = \max_{u(n-1) \in U(n-1)} \left\{ E_v \left\{ H\lfloor \bar{s}(n-1), u(n-1), n-1 \rfloor \bar{} + E^* \lfloor \bar{s}(n-1), n-1 \rfloor \bar{} \right\} \right\}$$

$$= \max_{u(n-1 \in U(n-1)} \left\{ \sum_{y=1}^{Y} P\lfloor \bar{r}(n-1), y \rfloor \bar{} \cdot \quad H\lfloor \bar{s}(n-1), u(n-1), \lfloor \bar{r}(n-1), y \rfloor \bar{}, n-1 \rfloor \bar{} + E^* \lfloor \bar{s}(n-1), n-1 \rfloor \bar{} \right\} \right\}$$

where $P\lfloor \bar{r}(n-1), y \rfloor \bar{}$ is the probability of the y-th discrete value of the random disturbance r affecting the system in the time interval beginning at n - 1. Conditional probability is introduced by replacing the independent probability density in equation (12) by the conditional probability density but the summation in equation (12) must be performed twice or more depending on the density introduced.

It can be seen from the above forward algorithm that such items as aquifer recharge, irrigation activities, reservoir release can be accommodated in the DP procedure. These events of the past influence the current time activities and can be accounted for in the procedure. This procedure has flexibility for aqueduct planning, storage design, treatment design, multiple-purpose reservoirs, branching multi-stage water resource systems, and for conjunctive surface and aquifer water use problems.

The advantages of DP have been pointed out in the generalization of the procedure to the recursive equation. These are flexibility to handle multi-

stage processes, stochastic disturbances, and the incorporation of constraints

Two serious disadvantages of DP are the dimensionality problem and the non-linear problem. In actuality, the objective function in DP can readily incorporate non-linearity but this leads to additional computational and dimension requirements. The dimensionality problem referred to is one of large high-speed computer memory requirement in the computation of DP using either the forward or backward algorithm. To give an idea of the problem, to solve the recursive equation (9) one must have at least ready access to storage locations associated with the terms:

$E^*\left[\bar{s}(n),\ n\right]$ for all lattice points

$s(n) \in S(n)$

$E^*\left[\bar{s}(n-1),\ n-1\right]$ for all lattice points

$s(n-1) \in S(n-1)$

$U^*\left[\bar{s}(n-1),\ n-1\right]$ for all lattice points

$s(n-1) \in S(n-1)$

If $s(n)$ and $s(n-1)$ are segmented into d_i levels in coordinate i, $i = 1, 2, \ldots, m$, then the total storage, d_t, required for the above terms becomes:

$$d_t = 3 \prod_{i=1}^{m} d_i$$

This storage requirement grows geometrically with m, the state domain dimension.[2/] As an example[3/], a water resource system consisting of four reservoirs ($m = 4$), and if each state is segmented into 15 levels ($d_i = 15$, for $i = 1, 2, 3, 4$), then at least a total storage memory of $3 \cdot 15$ units is required.

2/ L. M. Eisgruber has worked out the storage requirements for the forward algorithm of this type.

3/ This example was provided by V. T. Chow.

Computer time then becomes a problem. However, it must be noted that DP compared to direct enumeration increases in time saved as the number of stages increases. (Bellman and Dreyfus [3]). The time savings by DP are impressive but still much time is required in the steps to compute the decisions for each lattice point, retrieval to obtain the second right-hand term of the recursive equation, and to make the additions and compare with the results of the previous set of decisions. Evaluation of the objective function is done $N \cdot d_i^{1+m}$ times. Compared to $d_i^{n \cdot 1+m}$ times for direct enumeration the savings are significant but still hours rather than seconds are involved in the amount of time spent in central processing.

III. Discrete Differential Dynamic Programming

Some techniques have been developed to help remedy the dimensionality and non-linear problems associated with DP. Larsen [15] and Korsak and Larsen [13] and Larsen and Kechler [14] have presented an algorithm for generalizing Bellman's [2] successive approximation approach. This procedure has been demonstrated to be quite efficient both with respect to computer memory and time.

Differential dynamic programming is also a successive approximation procedure for reducing dimensionality and determining optimal policy in non-linear systems. Consider Bellman's partial differential equation of optimality for continuous systems:

$$(13) \quad \partial E/\partial t[s(t), t] = \max_{u(t) \in U(t)} \{H[s(t), t] + \langle E_s^*[s(t), t], \phi[s(t), u(t), t] \rangle\}$$

$E_s^*[s(t), t]$ is the gradient of the function $E^*[s(t), t]$ which is equal to $(\partial E^*/\partial s_1, \partial E^*/\partial s_2, \ldots, \partial E^*/\partial s_m)$. The notation $\langle E_s^*[\,,\,], \phi[\,,\,] \rangle$

signifies the scalar product of the two vectors E*, ϕ, having m components. Equation (13) may be solved given a boundary condition to determine $E^* \lfloor s(t), t \rfloor \ \forall \ t \in \lfloor t_o, t_e \rfloor$ by a procedure originally suggested by Bryson and Ho [4].

Consider also a continuous system with a set of non-linear time-varying differential equations:

$$\dot{s} = \phi \lfloor s, u, t \rfloor, \tag{14}$$

where s, u, and t are defined the same as for our review of DP above, ϕ is an m-dimensional vector functional, and $\dot{s} = ds/dt$.

$$E \lfloor s(t_e), t_e \rfloor = \left\{ \int_{t_o}^{t_e} H \lfloor s(p), u(p), p \rfloor dp \right. \tag{15}$$

be a performance criterion which is to be maximized. $E \lfloor s(t_e), t_e \rfloor$ is the sum of the returns due to transforming the system from an initial state at time t_o to a final state at time t_e. p is, in a sense, a dummy variable representing time. Equation (15) is constrained by the constraint set:

$$\begin{aligned} &s(t) \in S(t) \\ &u(t) \in U(t) \\ &t \in \lfloor t_o, t_e \rfloor \end{aligned} \tag{16}$$

Let us assume a trial policy $\underline{u}$ (t), $t \in \lfloor t_o, t_e \rfloor$ which is a decision vector satisfying equation set (16). Introduction of this policy into equation (14) provides a trial state trajectory $\underline{s}(t)$, $t \in (t_o, t_e)$, which also must satisfy equation set (16). The return from the system $\underline{E}(s, t_o)$ calculated from equation (15) may not be optimal. When a policy such as $\underline{U}(t)$ is applied, the system can only occupy the states defined by the state trajectory $\underline{s}(t)$. If the policy is permitted to vary by $\Delta u(t)$, $t \ \lfloor t_o, t_e \rfloor$ then a new policy:

$$u(t) = \underline{u}(t) + \Delta u(t) \tag{17}$$

is calculated which influences the state trajectory through equation (14),

and the new trajectory may be written:

$$(18) \quad s(t) = \underline{s}(t) + \Delta s(t).$$

The introduction of equations (17) and (18) into equations (13), (14), and (15) produces:

$$(19) \quad \frac{d}{dt}(\underline{s} + \Delta s) = \phi[\bar{s} + \Delta s, u + \Delta u, t\bar{]}$$

$$(20) \quad \max \underline{E}[\bar{s}_e, t_e\bar{]} = \int_{t_o}^{t_e} H[\bar{s} + \Delta s, \underline{u} + \Delta u, t\bar{]}dt$$

$$(21) \quad \frac{-\partial E^*[\bar{s} + \Delta s, t\bar{]}}{\partial t} = \max_{\Delta u} \left\{H[\bar{\underline{s}} + \Delta s, \underline{u} + \Delta u, t\bar{]} + \langle E^*_s[\bar{\underline{s}} + \Delta s, t\bar{]}, \phi[\bar{s} + \Delta s, \underline{u} + \Delta u, t\bar{]}\rangle\right\}$$

Jacobson [10] has expressed $E^*[\bar{\underline{s}} + \Delta s, t\bar{]}$ by a power series expansion with respect to $\underline{s}$. Substitution of these expansions and $E^*[\bar{\underline{s}}, t\bar{]} = \underline{E}[\bar{\underline{s}}, t\bar{]}$ + difference between optional trajectory $s^*(t)$ and trail trajectory, into equation (21) gives:

$$(24) \quad \max_{\Delta u} \left\{H[\bar{\underline{s}} + \Delta s, u + \Delta u, t\bar{]} + \langle(E^*_{\underline{s}} + E^*_{ss}\,\Delta s + E^*_{ss.....s}\,\Delta s), \phi[\bar{\underline{s}} + \Delta s, \underline{u} + \Delta u, t\bar{]}\rangle\right\}$$

where $E^*_{ss}\,\Delta s$ and $E_{ss.....s}\,\Delta s$ are the quadratic and higher order terms respectively. The solution of equation (24) will provide Δu^*, $t \in (t_o, t_e)$, the optimal increment at time t to obtain the optional policy. Jacobson [10] has also pointed out the possible infinite computing time and storage requirements for the parameters of the power series expansion, and has proposed truncation of all higher order terms except the quadratic term. Altorithms have subsequently been developed to obtain solutions if Δs is small enough to make the influence of higher order terms on the solution negligible.

This process reduces the global optimization of equation (24) to a local optimization or an optimization in the neighborhood of the trial

trajectory. If this trajectory is again improved within that neighborhood, smaller and smaller neighborhoods contain the new trial trajectories for each improvement and eventual convergence on the optimal trajectory takes place. The algorithms optimize the Hamiltonian in the neighborhood of the trial trajectory for each new improvement. This step reduces the infinite storage requirement to a manageable level.

Most empirical problems involving the search for an optimal policy involve discrete increments. The objective function is then like equation (4) for the system expressed in equation (1). The objective function is then to be maximized subject to the constraint set, equation (2). Let the initial and final m-dimensional state vectors be specified as $s(o) = a(o)$ and $s(N) = a(N)$ respectively.

Discrete differential dynamic programming (DDDP) then uses the differential approach just described to obtain a trial sequence of admissible decision vectors, $\underline{u}(n)$, $n = o, 1, \ldots, N - 1$, which satisfy equation (2), and the state vectors at different stages are subsequently determined. The sequence of values of the state vector satisfying equation (2) and the initial and final state vectors is the trial trajectory, $\underline{s}(n)$, $n = o, 1, \ldots, N$. This is used to calculate the trial policy $\underline{u}(n)$, $n - 0, 1, \ldots, N - 1$.

$\underline{u}(n)$ is introduced into equation (4) and:

$$(25) \quad \underline{E} = \sum_{n-1}^{N} H\left[\underline{s}(n-1), \underline{u}(n-1), n-1\right]$$

is obtained. This latter equation is the total return of the trial policy over the entire time horizon. One such $\underline{E}$ may or may not be the optimum return. A set of incremental states $\Delta s_{i,j}(n)$, $i = 1, 2, \ldots, T^m$, $j - 1, 2, \ldots, m$, can be specified from assumed incremental values of the j-th state domain and a fixed number, T, of incremental values can be considered at each stage. The number of Δs_i vectors at each stage is T^m. When added to the

trial trajectory at any stage, these vectors form an m-dimensional sub-domain, denoted as $\underline{s}(n) + \Delta s_i(n)$, $i = 1, 2, \ldots, T^m$. All sub-domains together form a "corridor" which is illustrated in Figure 1 for m = 1 and T = 3.

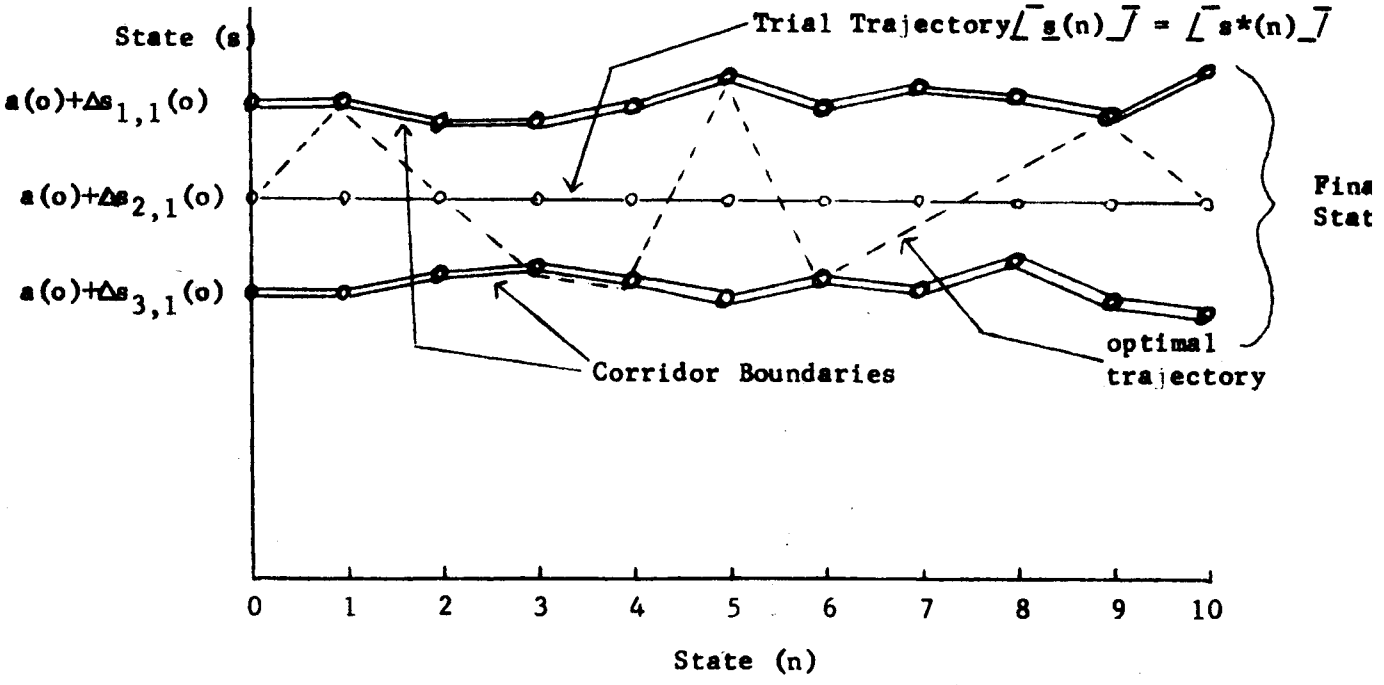

FIGURE 1: Trial Trajectory, m = 1, T = 3

The use of DDDP involves the corridor as the set of admissible states and the optimization is obtained by employing the recursion equation (9) previously formulated. The value of the return, E, obtained is at least equal to or greater than $\underline{E}$ from equation (25). If $E > \underline{E}$, the corresponding trajectory and policy obtained from the corridor are used in the next iteration step as the new trial trajectory and trial policy.

The value of such a technique in water resource system planning is now apparent. Various structures, flows, releases, and water uses can now be

accounted for by the state vectors and the stages of the DDDP system. Time and structure policy can be evaluated in terms of the returns from certain state transformations influencing optimal policy. Intermediate steps in the process of obtaining optimal policy can be retrieved since trial trajectories are used as initial improvement steps in the solution of the recursive relation presented above in equation (9). The corridor also constrains the decisions for each stage, thus cutting computer storage time and iterations involved in the solution.

IV. Decomposition of Dynamic Programming by Geometric Programming

Geometric programming (GP) is a rather simple technique used to optimize a set of positive polynomials subject to a constraint set. In general, the GP problem is concerned with:

$$\begin{aligned} &\min \; g_o(u) \\ &\text{subject to: } g_k(u) = 1, \; k = 1, \ldots\ldots, P \end{aligned} \tag{26}$$

where in general, $g = \sum_{i=0}^{P} Z_i$ and $Z_i = c_i \prod^{J} U_j^{a_{ij}}$ for $i = 1, 2, \ldots\ldots, n$, $j = 1, 2, \ldots\ldots, m$. Z_i are positive polynomials; c_i are positive coefficients; U_j are positive decision variables.

We make use of the geometric inequality; i.e., in our case for the system (26),

$$\sum_{i=1}^{n} b_i Z_i \geq \prod^{i} Z_i^{b_i} \tag{27}$$

expresses the geometric inequality, where $\sum_{t=1}^{n} b_i = 1$. Let $z_i = b_i Z_i$ for $i = 1, 2, \ldots\ldots, n$, then the inequality becomes:

$$\sum_{t=1}^{n} z_i \geq \prod^{i} v_i^{b_i} \tag{28}$$

where $v_i = (z_i/b_i)^{b_i}$. The left-hand side of inequality (28) is the primal

function g, and the right-hand side is the pre-dual function D. Since $Z_i = C_i \prod_j U_j^{a_{ij}}$, we can write the right-hand side as follows:

$$(29)\quad D(b, Z) = \left(\frac{c_1}{b_1}\right)^{b_1} \left(\frac{c_2}{b_2}\right)^{b_2} \cdots\cdots \left(\frac{c_n}{b_n}\right)^{b_n} U_1^{w_1} U_2^{w_2} \cdots\cdots U_n^{w_n}$$

where $W_j = \sum_{i=1}^{n} b_i a_{ij}$, $j = 1, 2, \ldots\ldots, m$. If the weights are chosen such that all $W_j = o$, then does not depend on the variables U_j and D is termed the dual function,

$$(30)\quad D(b) = \left(\frac{c_1}{b_1}\right)^{b_1}\left(\frac{c_2}{b_2}\right)^{b_2} \cdots\cdots \left(\frac{c_n}{b_n}\right)^{b_n},$$

with weights satisfying $w_j = \sum_{i=1}^{w} b_i A_{ij} = o$, $j = 1, 2, \ldots\ldots, m$. These weights can be found by the system,

$$\underline{A}^{t}\underline{b} = 0$$

$$b_i \geq 0, \quad \sum_{i=1}^{n} b_i - 1$$

where $\underline{A}$ is the matrix of the exponents, A_{ij}.

This formulation follows the derivation of the dual function by Duffin, et al [7], and they further show that the optimizing values of U_j, the decision variables; are given by $U_j = b_i D(b)$ for $i = 1, 2, \ldots\ldots, n$, $j = 1, 2, \ldots\ldots, m$.

The DP problem can be decomposed to finite subproblems which become GP subproblems. This is the main feature of the GP technique. The n initial and ending stage returns and decisions are similarly related through the successive (forward algorithm) GP subproblems. In many cases each GP subproblem is reduced to simple land calculations and at most very straightforward calculations need be made. Depending on this reduction, the computer memory involved in GP calculation is reduced substantially from the original

multi-stage DP algorithm discussed above, and may be substantially reduced from the memory required to deal with the DDDP problem. Actually, the U_j and b_i tables one gets as intermediate calculations in any particular GP subproblem specifies the corridor for future stages. This is the same as the feasible decision set corridor of the DDDP procedure except specific polynomial functions for $g_o(u)$ and $g_k(u)$ are required.

V. An Example Using GP: Water Quality Design

To illustrate the ease of using GP in each stage of a DP problem let us consider a multi-stage water quality design problem. Consider a stream along which three communities are located, all discharging wastewater without treatment. Suppose the communities are cooperating with a state agency which is charged with the responsibility of pollution control and the abatement actions to be taken by each community which will result in the attainment of at least some minimum determined stream quality standards. These standards are to be met by each community at least total annual cost to all three communities and to the state since the better agency is involved in some form of sharing of treatment costs. Assume the water quality standards have been set based upon the concentration of BOD in the stream at the top of each reach of the stream before entering community use intakes. The concentration is measured in mg./l. per time unit (Klein [12].)[4/]

Assume a mix of pollution control methods has been determined to be advisable. The problem then becomes one of designing a treatment system for the three communities at minimum cost. Also, the design must allow water

---

4/ No allowance is made for variation in stream mixture and variation over the reach of the stream from one community to another. The oxygen-say equation may be introduced but is not specified in the example.

treatment in accordance with the BOD concentration standards. This problem, one of multi-stage nature, can be dealt with by use of the recursive relation and the forward algorithm of DP. The stream reaches between community intake differentiate the stages involved. The recursive relation of DP now becomes:

$$(31)\quad E_n^*(s) = \min_U \left[E(s)_{U_n},\ E^*_{n-1}\ (U_n) \right]$$

where $E_n^*(s)$ is the value of the optimal policy when in state, s, and these are n stages remaining. $E(s)_{U_n}$ is the immediate value for the return associated with decision, U_n, for j decision variables, in state s with n stages remaining. $E^*_{n-1}(U_n)$ is the total value associated with optimal policy when n - 1 stages remain and given decision U is made for j decision variables.

For the three-community water quality use the recursive relation can be specified as:

$$(32)\quad E^*(L_i/T) = \min_{Li=1/B} \left[D(b)_{Li/t} + E^*(L_i\text{-}1/B) \right]$$

where Li/T specifies the BOD concentration in liters per time unit in Klein's [12] notation at the top of reach i. $Li/B = Li/T^{e^{-kt}}$ is the BOD concentration at the bottom of reach i. k is the deoxygenation constant. Li-1/B is the concentration of BOD at the bottom of reach i - 1. $D(b)_{Li/T}$ is the dual value of the GP subproblem i, where in (32) the recursive relation has been decomposed to i = 1, 2, 3 GP subproblems.

The BOD concentration at the top of reach i, Li/T, is calculated by:

$$(33)\quad Li/T = \frac{Lo/B\ \ V_R + E_i\ V_{E_i}}{V_R + V_{E_i}}$$

where, Lo/B is the BOD concentration just upstream from the top of reach i, or is a bottom reach concentration from some other system. V_R is the minimum average 7-day flow occuring once in 10 years (MAF7/10). E_i is the waste water discharge of the i-th community in mgd. V_{Ei} is the BOD concentration of the waste water of the i-th community.

Three treatment techniques; e.g., treatment, process change, and ponding, result in U_1, U_2, U_3 percent of the influent BOD remaining upon completion of each named technique, respectively, when applied to the influent. Suppose the maximum permissable level of BOD in the stream at the top of reach i = 1 is 5 mg./l., then the multiple U_1, U_2, U_3 must satisfy, Lo/B V_R + E, VE, $U_1\ U_2\ U_3/V_R + V_{E_1} \leq$ t mg./l. = $\bar{L}_i/K$. This is the constraint the BOD standard imposes on the system at reach 1. Consider, for example, the following data for the three communities involved.

Community 1:

Lo/B = 1.0 mg./l.

V_R - 50 mgd

V_{E_1} = 5 mgd

E_1 = 200 mg./l.

K = .3

t = .3 (deoxygenation constant)

t = 1.0 (time of flow)

$L_1/t \leq 5.0$ mg./l.

$Cost_1 = 50\ U_1^{-1.3} + 20\ U_2^{-1.4} + 10\ U_3^{-1.5}$ (thousand $/year)

Community 2:

E_2 = 220 mg./l.

V_{E_2} = 10 mgd.

R^2 = .2

t = 1.1

$L_2/t \leq 4.0$ mg./l.

$cost_2 = 30\ U_1^{-1.3} + 10\ U_2^{-1.4} + 40\ U_3^{-1.5}$

Community 3:

$E_3 = 250$ mg./1.

$V_{E_3} = 12$ mgd.

$K = .2$

$5 = 1.5$

$L_3/T \leq 5.0$ mg./1.

$\text{cost}_3 = 60\ U_1^{-1.3} + 50\ U_2^{-1.4} + 70\ U_3$

We now can incorporate the quality standard constraint with the criterion function to obtain solutions to each GP subproblem, which, once obtained is a preliminary design for the three-community water quality system. The solutions for the U_1, U_2, U_3 treatment techniques specify the mix of technique which should be followed at each reach (stage) along the stream in order to obtain a least cost design.

If we require $U_1\ U_2\ U_3 \leq .225$ or $4.45\ U_1\ U_2\ U_3 \leq 1.0$, then subproblem 1 (the first community) is then:

$$\text{Min } 50U_1^{-1.3} + 20U_2^{-1.4} + 10U_3^{-1.5}$$

(34)

$$\text{S.T.} \quad 4.45U_1\ U_2\ U_3 \leq 1.0$$

$$U_1, U_2, U_3 > 0$$

Using the GP procedure above to solve for the weights, b_i, $b_1 = .358$, $b_2 = .332$, $b_3 = .310$, $b_4 = .465$. The dual function becomes:

$$(35) \quad D(b) = \left(\frac{50}{.358}\right)^{.358} \left(\frac{20}{.332}\right)^{.332} \left(\frac{10}{.31}\right)^{.31} (4.45)^{.465} = \$134{,}000$$

Now let us vary L_1/T for a range of feasible values $0 < L_1/T \leq \bar{L}_i/T = 5.0$ mg./1 (The BOD upper limit standard). If we choose 4.0 mg./1., 3.0 mg./1., 2.0 mg./1., and 1.0 mg./1., we get the following constraints:

$$5.88\ U_1\ U_2\ U_3 \leq 1$$

$$8.7\ U_1\ U_2\ U_3 \leq 1$$

$$16.7\ U_1\ U_2\ U_3 \leq 1$$

$$200.0\ U_1\ U_2\ U_3 \leq 1$$

The dual functions for each of these choices of L_1/T varies also as,

D(b) for 4.0 mg./1. = \$153,000

" " 3.0 " " = \$183,000

" " 2.0 " " = \$248,000

" " 1.0 " " = \$788,000

Note that the optimal values of the U_2 are not yet chosen. The costs are expressed solely in terms of stream quality. Once an optimal level of stream quality is chosen, the U_j can then be determined giving us the necessary information for structure and treatment design.

The quality at the bottom of the first reach will vary as we vary the quality of the water at the top of the reach, thus affecting the second subproblem. For the five quality standards chosen at the top of the reach, the following $L_1/B = L_1/T\ e^{-kt}$ are obtained:

$5e^{-.3} = 3.7$ mg./1.

$4e^{-.3} = 2.96$ mg./1.

$3e^{-.3} = 2.22$ mg./1.

$2e^{-.3} = 1.48$ mg./1.

$1e^{-.3} = .74$ mg./1.

Any one of these quality levels is used to initialize the second subproblem. Likewise, different water quality levels at the bottom of the second reach will affect the third subproblem.

Generalization to many stages, or reaches in the case of the above sketch of the water quality problem, requires necessary computer time but the software and memory are minimal compared to DP. Also, the nonlinearity introduced

in the problem above, or any problem, is easily decomposed in the dual function using the geometric inequality, assuming proper weights, b_i, can be found.

The basic multi-stage planning technique of DP is preserved in both the DDDP and GP procedures. However, the severe computational problems of DP become modified with the use of the latter two procedures. Nonlinearities and stochastic disturbances may also be incorporated in DDDT as well as GP if each stage is decomposed to a given domain of the appropriate probability density. The future use of DDDP and GP in water resource planning looks encouraging since all the facets of the DP technique can be handled by the modified procedures and computations made in much less time and with considerable reduction in computer memory.

BIBLIOGRAPHY

1. Bellman, R., Dynamic Programming, Princeton: Princeton University Press, 1957.

2. _____, Adaptive Control Processes, Princeton: Princeton University Press, 1961.

3. _____, and S. E. Dreyfus, Applied Dynamic Programming, Princeton: Princeton University Press, 1962.

4. Bryson, A. E., and Y-C. Ho, Applied Optimal Control, Waltham, Massachusetts: Blaisdell Publishing Co., 1969.

5. Burt, O. R., "The Economics of Conjunctive Use of Ground and Surface Water," Hilgardia, Vol. 36, No. 2, University of California, Division of Agricultural Sciences (December, 1964).

6. Chow, V. T., and D. D. Meredith, "Review of Programming Techniques", Water Resources Systems Analysis, Part IV, Hydraulic Engineering Series No. 22, Civil Engineering Studies, University of Illinois, Urbana, Illinois (July, 1969).

7. Duffer, R. J., E. L. Peterson, and C. Zener, Geometric Programming -- Theory and Applications, New York: John Wiley, 1967.

8. Gablinger, M., and D. P. Loucks, "Markov Models for Flow Regulation," Journal of the Hydraulics Division, American Society of Civil Engineers, Vol. 96, (January, 1970).

9. Hall, W. A., R. C. Garboe, W. W.-G. Yeh, and A. M. Askew, Optimum Firm Power Output from a Two Reservoir System by Incremental Dynamic Programming, Contribution No. 130, Water Resources.

10. Jacobson, D. H., "New Second-Order and First-Order Algorithms for Determining Optimal Control: A Differential Dynamic Programming Approach," Journal of Optimization Theory and Applications, Vol. 2, No. 6 (June, 1968).

11. Kerris, K. D., "An Economic Approach to Water Quality Control" Journal of the Water Pollution Control Federation (December, 1966).

12. Klein, L., River Pollution, London: Butterworth and Company, 1966.

13. Korsak, A. J., and R. E. Larson, "A Dynamic Programming Successive Approximation Technique with Convergence Proofs, Part II: Convergence Proofs," IFAC Automatica, Vol. 6, No. 2 (March, 1970).

14. Larson, R. E., and W. G. Keckler, "Applications of Dynamic Programming to Water Resources Problems," paper presented to IFAC Haifa Conference on Computer Control of Natural Resources and Public Utilities, Haifa, Israel, (September, 1967).

15. Larson, R. E., State Increment Dynamic Programming, New York: American Elsevier Publishing Company, 1968.

16. Maas, A., M. M. Hufschmidt, R. Dorfman, H. E. Thomas, S. A. Marglin, and G. M. Fair, Design of Water Resources Systems, Cambridge: Harvard University Press, 1962.